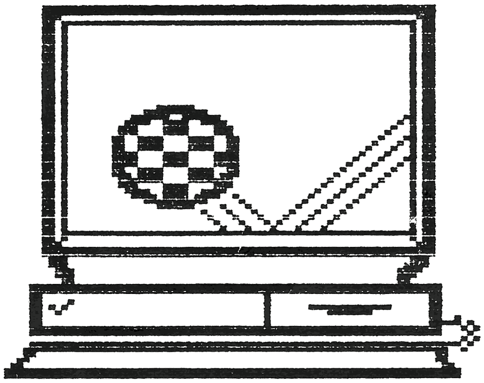


AMIGA MAIL™

July '86

AMIGA DEVELOPER CONFERENCE

Thanks for your responses to the Amiga Developers Conference inquiry. We have received a lot of good ideas and suggestions which we will use. Plans are currently underway. We should have date, location, and pricing information available in the next issue of AmigaMail.
(HINT: don't make any plans for the middle of September!)
Thanks again for your input. We will keep you posted...

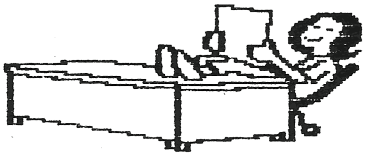
POLICY CHANGE:

The Amiga documentation is now available from Addison-Wesley and Bantam publishers. They are being sold by authorized Amiga dealers and bookstores. Due to its easy access, Commodore will no longer supply them to developers.

Please take note that: Commodore will not directly sell the Amiga developer's kit or documentation. This is a change from last month's newsletter. All existing orders will be filled, however no new orders will be taken.

Product Information

Due to the number of developer's sending product information for mention in AmigaMail, it would simplify matters if the developer would contact Becky Cotton before submitting information. We at AmigaMail would like to ensure that products mentioned in the newsletter are indeed available. If you have a product available for the Amiga, let us know, we will be glad to make note of it in our newsletter. If interested, contact Becky Cotton at (215) 431-9180.



? ? Questions & Answers ? ?

What are the synthesizer capabilities of the Amiga?

The Amiga has four 8 bit D/A converters. Each D/A has a programmable (by the 68000) volume and sampling rate (period) control. They are wired into 2 low-pass filters for the left and right stereo channels. Each D/A converter can be loaded by the 68000 or by it's own DMA channel. Each audio DMA channel can be used either to feed samples to it's D/A converter or to set the volume and/or period of another D/A converter. The DMA channels automatically restart at the end of the buffer.

specs:

DMA channels-

addressing range: 0-0x7FFFF (word aligned)
buffer size range: 2-131072 samples (even)
maximum range: 28867.3 samples per second

D/A converters-

amplitude range: -128 thru 127
amplitude accuracy: < 1/2 least signif. bit
sampling rate range: 279.365 ns-18.3085 ms
sampling rate resolution: 279.365 ns
volume control range: 0 thru 64 linear

low-pass filters-

slope: 6 pole, 36db/octave
pass band response: + or - 3db 15hz-4kHz
stop band response: -40db at 7kHz

SOFTWARE:

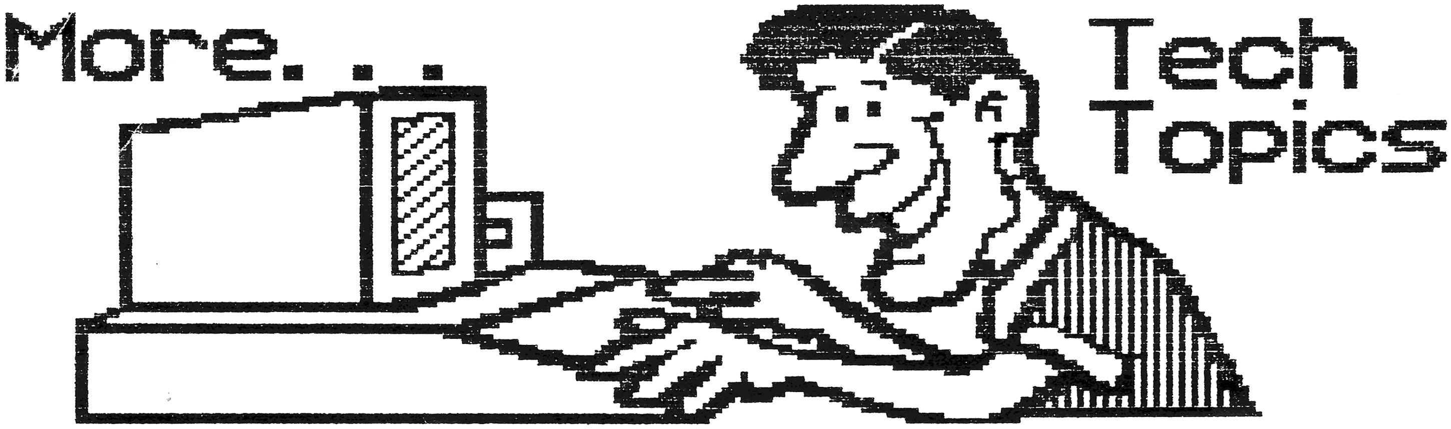
ROM Kernal audio is an exec device. Commands include allocation and arbitration (prioritize stealing) of channels, and DMA channel control.

Sam Dicker
Commodore-Amiga

If you have a question regarding Amiga development which could be addressed in our newsletter, send it to:

Becky Cotton
Software Department
Commodore Business Machines
1200 Wilson Drive
West Chester, PA 19380

More... Tech Topics



Important Notes:

For all developers who have purchased the Hardware Schematics and Expansion Specifications:

1. This is not a change, but it is an important detail that some developers have missed up to now.

On the schematic of the backplane, please note that:

SLAVEIN* comes in on pin 11 of the upstream 86 pin connector.

SLAVEOUT* goes out on pin 11 of the downstream 86 pin connector.

As stated above, this is not a change, however some developers were not sure on which pin #'s the SLAVE signals were connected.

2. We have recently discovered that in a 5 board backplane, the termination resistors slightly overload the BAS* driver. If you are designing a 5 board backplane, we recommend that you change the values of these resistors to 390 ohm (down) and 270 ohm (up) on the BAS* line.

If you are designing a backplane with MORE than 5 boards, you need to do a loading analysis of all of the terminated signals on the backplane. On those signals which are overloaded, increase the value of the resistors, do NOT just use stronger drivers in place of the 74LS245s because other boards plugged into your backplane will try to drive your bus with 74LS245s.

Commodore-Amiga

```
Software Failure. Press left button.
  Guru Meditation #0000004.00018E8
```

Look familiar? When you get alerts in the form 0000000#., you've taken a 68000 exception, as follows:

2	bus error
3	address error
4	illegal instruction
5	division by zero
8	privilege instruction
9	trace (single step)
A	line 1010 emulator
B	line 1111 emulator

SYNTHESIZED SPEECH

The SAY command converts text to synthesized speech. It is possible to enter text and command interactively. There are many options to be selected with SAY. To select the options, type them in when the Input window is selected. Next, press RETURN. The options are:

-m	male voice
-f	female voice
-r	robot voice
-n	natural inflection
-s##	speed (i.e. -s200 from 40-400)
-p##	pitch (i.e. -p200 from 65-320)
-x	filename

It is possible to enter text and/or command into a file, then execute the file using the -x option.

PRINTER CAPABILITY LISTING

Color Printers

Printer	Driver	Comments
Canon PJ1080A	Canon PJ1080A	Requires Custom Driver 3
Diablo C150	Diablo_C150	
Epson JX-80	Epson_JX80	
Apple Image Writer II	Image Writer II	Requires Custom Driver 2
Juki 5510	Epson_JX-80	Requires a Color Upgrade
NEC CP2 & CP3	Epson_JX-80	
Okimate 20	Okimate_20	Requires Amiga Plug "N" Print Module

Dot Matrix Printers

Printer	Driver	Comments
Alphacom Alphapro	Alphacom_Alphapro_101	
Apple Image Writer	Image Writer	Requires Custom Driver 3
Brother 1509	Epson	
Citizen MSP Series	Epson	
C. Itoh 8510	Pro Driver	Requires Custom Driver 5
CIE CI-3500	CBM_MPS1000	Req. IBM Compatible Module
Epson FX Series	Epson	
Epson LX Series	Epson	
Epson MX Series	Epson	
Epson RX Series	Epson	
IBM Graphics (5152)	CBM_MPS1000	
Juki 5510	Epson JX-80	
Okidata 182 & 183	CBM_MPS1000	Set in IBM Compatible Mode
Okidata 192 & 193	CBM_MPS1000	Set in IMB Compatible Mode
Okidata 292 & 293	CBM_MPS1000	Req.IBM Plug"N"Print Module Does not support color 4
Panasonic 1080	Epson	
Panasonic 1091	Epson	
Panasonic 1092	Epson	
Panasonic 1592	Epson	
Radio Shack DMP 300	CBM_MPS1000	
Star SG Series	CBM_MPS1000	All Features not Implement.
Star SD & SR Series	CBM_MPS1000	All Features not Implement.
Star MX Series	Epson	Set in Epson Compat. Mode
	CBM_MPS1000	Set in IBM Compatible Mode
Star NB Series	CBM_MPS1000	Set in IBM Compatible Mode
Star NL Series	Epson	Req. Epson Compat. Module
	CBM_MPS1000	Req. IBM Compat. Module

Daisy Wheel Printers

Printer	Driver	Comments
Brother HR-15XL	Brother_HR-15XL	
Citizen Premier 35	Diablo_630	
Comrex CR-II	Brother_HR-15XL	
Comrex CR-III	Brother_HR-15XL	
Commodore 6400	Diablo_630	Requires Centronics Opt.
Diablo Advantage D25	Diablo_Advantage D25	
Diablo 630	Diablo_630	
Dynax	Brother_HR-15XL	
Epson DX-20	Diablo_630	
Juki 6000	Diablo_630	
Juki 6100	Diablo_630	
Juki 6300	Diablo_630	
Qume LetterPro 20	Qume_LetterPro 20	
Qume Sprint II	Diablo_630	
Panasonic KXP 3131	Diablo_630	
Panasonic KXP 3151	Diablo_630	
Tec F10	Diablo_630	

Laser-Jet Printers

Printer	Driver	Comments
Canon Laser	Epson	Set in FX-80 Mode
CIE Lips 10	Epson	Set in Epson Mode
	Diablo_630	Set in Diablo 630 Mode
H.P. Laser Jet	H.P._Laser_Jet	
H.P. Laser Jet Plus	H.P._Laser Jet_Plus	

NOTE:

1 -All of the printers listed above require the appropriate printer cable.

2 -Redmond Cable has released an Apple Image Writer Driver which they are selling with the cable. There is a version for the Image Writer and the Image Writer II. Redmond Cable may be contacted at (206) 868-2168.

3 -Contact Tycom Technologies at (216) 261-7088.

4 -V.1.2 Workbench will inclued a complete Oki 292 Driver.

5 -$20.95 From: Micro-Cybernetics Corp.
P.O. Box 3126
Laurel, MD 20708

6 -Unsupported printers may not be totally compatible.

```
/***********************************************************************
 *  LatFFP -- Program to show the use of Motorola Fast Floating Point  *
 *            libraries from V1.1 release of Lattice C for the Amiga.  *
 *                                                                     *
 *  Larry Hildenbrand (11/04/85) originally wrote the program to show  *
 *  the use of  ALL the  Motorola Fast  Floating  Point routines with  *
 *  V1.0 Lattice C for the Amiga in comparison to the DOUBLE precision *
 *  IEEE floating point math which is built into the C.                *
 *                                                                     *
 *  Lisa Siracusa (07/23/86) modified the program to show the kludge   *
 *  that is needed in order  to  access  the math FFP routines.  The   *
 *  variables used  for FFP  math are   defined as ints.  You can not  *
 *  use FLOATs because V1.1 Lattice C converts  FLOAT to DOUBLE dur-   *
 *  ing  expression  evaluation  and when passing arguments.  Only a   *
 *  few of the routines are shown below.   This will  give a  general  *
 *  idea of what must be done in order to access the routines.         *
 ***********************************************************************/

#include ":include/exec/types.h"
#include ":include/lattice/math.h"


/* ??? #define   E      2.718281828459045 */   /* V1.0 Lattice C BUG!   See */
#define   PIME    0.423310826                   /* below.  PIME == PI - E.   */


extern   int      SPAdd();
extern   int      SPSub();
extern   int      SPMul();
extern   int      SPDiv();

extern   int      SPAtan();
extern   int      SPSin();
extern   int      SPCos();
extern   int      SPTan();
extern   int      SPSincos();
extern   int      SPExp();
extern   int      SPLog();
extern   int      SPPow();
extern   int      SPSqrt();
extern   int      SPTieee();
extern   int      SPFieee();

char st1[80] = "3.1415926535897";
char st2[80] = "2.718281828459045";

int MathBase;         /* Basic FFP lib pointer          */
int MathTransBase;    /* Transcendental FFP lib pointer */

int dots_good = 0;

union kludge1         /* Can't use FLOAT directly for FFP stuff */
{                     /* because V1.0 Lattice converts FLOAT to */
  FLOAT num1;         /* DOUBLE in expressions and when passing */
  int   i1;           /* parameters.                            */
```

```
} k1;

union kludge2
{
  FLOAT num2;
  int   i2;
} k2;

union kludge3
{
  FLOAT num3;
  int   i3;
} k3;

union kludge4
{
  FLOAT num4;
  int   i4;
} k4;

union kludge5
{
  FLOAT num5;
  int   i5;
} k5;

union kludge6
{
  FLOAT num6;
  int   i6;
} k6;



show_dot()
{
    if(++dots_good == 1000)
    {
        dots_good = 0;
        printf(".");
    }
}



show_result(num)
FLOAT num;
{
    printf("\nResult = %f", num);
}


show_result_ffp(in_val)   /* Convert to IEEE and display */
int in_val;
{
union kludge_sr
```

```
{
    FLOAT new_iv_f;
    int   new_iv_i;
} k;

    k.new_iv_i = SPTieee(in_val);
    show_result(k.new_iv_f);
}



main()                    /* Lattice/FFP test code */
{
UWORD i;

    printf("Motorola Fast Floating Point example using Lattice C for the Amiga\n

/****************************************************/
/*                                                  */
/*                Open FFP Libraries                */
/*                                                  */
/****************************************************/

    if((MathBase = OpenLibrary("mathffp.library",0)) < 1)
    {
       printf("\n\n*** ERROR ***  Can't open mathffp.library: vector = %lx\n", M
       exit();
    }
    else
       printf("\n\nSuccessfully opened mathffp.library: vector = %lx\n", MathBas

    if((MathTransBase = OpenLibrary("mathtrans.library",0)) < 1)
    {
       printf("*** ERROR ***  Can't open mathtrans.library: vector = %lx\n", Mat
       CloseLibrary(MathBase);
       exit();
    }
    else
       printf("Successfully opened mathtrans.library: vector = %lx\n", MathTrans


/****************************************************/
/*                                                  */
/*                 FFP SPAdd & SPSub                */
/*                                                  */
/****************************************************/

    k1.num1 = PI;                 /* V1.0 Lattice C BUG!  Can't have two  */
    k2.num2 = k1.num1 - PIME;     /* constant assignments in a row.  Fake */
                                  /* it by making the second be an        */
                                  /* expression!                          */

    k1.i1 = SPFieee(k1.i1);
    k2.i2 = SPFieee(k2.i2);
    printf("\n\n50,000 additions of %s to %s\n", st1, st2);
    for(dots_good = 0, i= 1; i < 50000; i++)
    {
       k3.i3 = SPAdd(k2.i2, k1.i1);
       show_dot();
    }
    show_result_ffp(k3.i3);

    printf("\n\n50,000 subtractions of %s to %s\n",st1,st2);
    for(dots_good = 0, i = 1; i < 50000; i++)
    {
        k3.i3 = SPSub(k1.i1, k2.i2);
        show_dot();
    }
    show_result_ffp(k3.i3);

/***************************************************/
/*                                                 */
/*              FFP SPMul & FFP Div                */
/*                                                 */
/***************************************************/

    k1.num1 = PI;
    k2.num2 = k1.num1 - PIME;

    k1.i1 = SPFieee(k1.i1);
    k2.i2 = SPFieee(k2.i2);

    printf("\n\n50,000 multiplies of %s by %s\n",st1,st2);
    for(dots_good = 0, i= 1; i < 50000; i++)
    {
       k3.i3 = SPMul(k2.i2, k1.i1);
       show_dot();
    }
    show_result_ffp(k3.i3);

    printf("\n\n50,000 divides of %s by %s\n",st1,st2);
    for(dots_good = 0, i = 1; i < 50000; i++)
    {
      k3.i3 = SPDiv(k2.i2, k1.i1);
      show_dot();
    }
    show_result_ffp(k3.i3);


/**************************************************/
/*                                                */
/*        FFP SPSqrt, SPLog, SPExp & SPPow        */
/*                                                */
/**************************************************/

    k1.num1 = PI;
    k2.num2 = k1.num1 - PIME;

    printf("\n\n50,000 square roots of %f\n",k2.num2);
    k1.i1 = SPFieee(k1.i1);
    k2.i2 = SPFieee(k2.i2);
    for(dots_good = 0, i= 1; i < 50000; i++)
    {
        k3.i3 = SPSqrt(k2.i2);
        show_dot();
    }
    show_result_ffp(k3.i3);

    printf("\n\n40,000 logarithms of %s\n",st1);
    for(dots_good = 0, i= 1; i < 40000; i++)
    {
        k3.i3 = SPLog(k1.i1);
        show_dot();
    }
    show_result_ffp(k3.i3);

    printf("\n\n40,000 exponents of %s\n",st1);
    for(dots_good = 0, i= 1; i < 40000; i++)
    {
        k3.i3 = SPExp(k1.i1);
        show_dot();
    }
    show_result_ffp(k3.i3);


/************************************************/
/*                                              */
/*        FFP SPSin, SPCos, SPTan, SPSincos &   */
/*        SPAtan                                */
/*                                              */
/************************************************/

    k1.num1 = PI;
    k2.num2 = k1.num1 - PIME;
    k1.num1 = k1.num1 / 6.0;
    k1.i1 = SPFieee(k1.i1);
    k2.i2 = SPFieee(k2.i2);

    printf("\n\n20,000 sines, cosines and tangents of %s / 6 radians\n", st1);
    for(dots_good = 0, i= 1; i < 20000; i++)
    {
        k2.i2 = SPSin(k1.i1);
        k3.i3 = SPCos(k1.i1);
        k4.i4 = SPTan(k1.i1);
        k5.i5 = SPSincos(&k6.i6, k1.i1);  /* V1.0 FFP BUG!  Cosine return  */
                                          /* value (k6.i6) of SPSincos is  */
                                          /* broken.  Function result      */
                                          /* (sine -- k5.i5) is OK.        */

        show_dot();
    }
    show_result_ffp(k2.i2);
    show_result_ffp(k3.i3);
    show_result_ffp(k4.i4);
    show_result_ffp(k5.i5);
    show_result_ffp(k6.i6);
    printf("\n\n20,000 arctangents of the tangent of %s / 6 radians\n", st1);
    for(dots_good = 0, i= 1; i < 20000; i++)
    {
        k2.i2 = SPAtan(k4.i4);
        show_dot();
    }
    show_result_ffp(k2.i2);


/**************************************************/
/*                                                */
/*             Close FFP Libraries                */
/*                                                */
/**************************************************/

    RemLibrary(MathTransBase);     /* mark lib for Expunge() from RAM upon  */
                                   /* CloseLibrary() by last opener.  Else  */
                                   /* lib stays in RAM for others to use    */
                                   /* quickly (no need to go to disk) until */
                                   /* AllocMem() finds it needs the memory  */
                                   /* for some other purpose.               */

    CloseLibrary(MathTransBase);   /* Close transcendental math RAM library */
    CloseLibrary(MathBase);        /* Close basic math ROM library */
    printf("\n\n\nEnd Lattice FFP Examples\n");
}
```

this document was
generously
contributed by

randell jesup

scanned by:
www.commodore.international